UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Modelos Avanzados de Computación

Práctica 5

Clases de complejidad

Curso 2014-2015

Doble Grado en Ingeniería Informática y Matemáticas

Práctica 5

Clases de complejidad

1. Demostrar que los siguientes problemas están en **P**:

   - **Exponenciación modular** [Modular exponentiation]: Dados 4 enteros a, b, c, p determinar si $a^b \equiv c(\mathrm{mod} p)$
   - **Grafos acíclicos**: Un grafo dirigido se dice que es *acíclico* si no tiene ciclos. Demuestre que todo grafo dirigido acícilo tiene una *fuente* (un nodo al que no llegan arcos). Demuestre que un grafo dirigido con n nodos es acíclico si y sólo si se pueden numerar los nodos del 1 al n de manera que siempre los arcos van desde números más pequeños a números más grandes. Finalmente, diseñe un algoritmo polinómico para determinar cuándo un grafo es acíclico.
   - **Grafos bipartitos**: Un grafo no dirigido es bipartito si sus nodos se pueden dividir en dos conjuntos, no necesariamente del mismo tamaño, de manera que todas las aristas del grafo conectan un nodo del primer conjunto con un nodo del segundo. Demuestre que un grafo es *bipartito* si y sólo si todos sus ciclos son de longitud par. A continuación, diseñe un algoritmo polinómico para comprobar si un grafo es bipartito.

2. Sea el problema *factorización* que consiste en dados dos números x, y determinar si x tiene un divisor k que sea $1 < k < y$. Demostrar que este problema está en **NP** $\cap$ **coNP**.

3. Determine las clases de complejidad a las que pertenecen los siguientes problemas:

   - 2-SAT

     Christos H. Papadimitriou (1994): Computational Complexity, theorem 16.3

   - Undirected st-Connectivity (a.k.a. Undirected Reachability or UST-CON): Dados dos vértices s y t en un grafo no dirigido, determinar si t es alcanzable desde s.

- st-Connectivity (a.k.a. Reachability or STCON): Dados dos vértices s y t en un grafo dirigido, determinar si t es alcanzable desde s.

  Christos H. Papadimitriou (1994): Computational Complexity, theorem 16.2

- Unique decipherability

  Wojciech Rytter (1986):
  "The space complexity of the unique decipherability problem"
  Information Processing Letters 23(1):1-3
  DOI 10.1016/0020-0190(86)90121-3

- Horn-SAT, con cláusulas de Horn
- El problema de la parada en T pasos [T-step Halting problem]: Dada una máquina de Turing y un número T (en unario), ¿se detiene la máquina en los primeros T pasos?
- El problema de la parada [Halting problem]
- Conjunto independiente maximal [Maximum independent set]: Dado un grafo G y un entero k, ¿tiene el conjunto independiente más grande de G un tamaño k?
- Número cromático $\chi(G)$ [Chromatic number]: Dado un grafo G y un entero k, ¿es k el menor entero tal que G sea k-colorable?
- Ajedrez: ¿Mate en k movimientos?

4. Determine, dentro de la jerarquía polinómica, a qué clase de complejidad pertenecen cada uno de los siguientes problemas:

   - MaxSAT (optimización): Dada una fórmula SAT ϕ, encontrar una asignación que satisfaga el número máximo posible de cláusulas de ϕ.
   - MaxSAT (valor): Dada una fórmula SAT ϕ, determine el número máximo posible n de cláusulas de ϕ que se pueden satisfacer simultáneamente.
   - MaxSAT (umbral): Dada una fórmula SAT ϕ y un umbral t, ¿existe una asignación que satisfaga t o más cláusulas de ϕ?

   [Moore & Mertens: "The Nature of Computation", 9.1, pages 352-353]

5. Una Máquina de Turing no-determinística fuerte es una máquina que tiene tres posibles respuestas 'Si', 'No', y 'Duda'. Una de estas máquinas decide L si y solo si, para para todo $x \in L$, todos los cálculos posibles terminan en 'Si' o 'Duda' y al menos uno en 'Si' y para todo $x \notin L$, todos los cálculos posibles terminan en 'No' o 'Duda' y al menos uno en 'No'. Demuestre que L es decidido por una Máquina de Turing no-determinística fuerte si y solo si L está en $\mathbf{NP} \cap \mathbf{coNP}$.

Christos H. Papadimitriou: "Computational Complexity", problem 10.4.6

6. Demuestre que la clase **P** es cerrada para la unión y la intersección.

Christos H. Papadimitriou: "Computational Complexity", problem 7.4.5

7. Sea C una clase de funciones de los enteros no negativos en los enteros no negativos. Se dice que C es cerrada para la composición polinómica por la izquierda si $f(n) \in C$ implica que $p(f(n)) = O(g(n))$ para alguna función $g(n) \in C$, y donde $p(n)$ es un polinomio cualquiera. Se dice que C es cerrada para la composición polinómica por la derecha si $f(n) \in C$ implica que $f(p(n)) = O(g(n))$ para alguna función $g(n) \in C$, y donde $p(n)$ es un polinomio cualquiera.

   Determine cuáles de las siguientes clases de funciones son cerradas para la composición polínómica por la derecha y cuáles lo son por la izquierda.

   a) $\{n^k : k > 0\}$
   b) $\{n \cdot k : k > 0\}$
   c) $\{k^n : k > 0\}$
   d) $\{2^{n^k} : k > 0\}$
   e) $\{\log^k(n) : k > 0\}$
   f) $\{\log(n)\}$

Christos H. Papadimitriou: "Computational Complexity", problem 7.4.4

8. Demuestre que $\mathbf{NP} \neq \mathbf{SPACE}(n)$.
   Tenga en cuenta que no se sabe si una de las clases está incluida en la otra.

Arora & Barak: "Computational Complexity: A Modern Approach", exercise 3.2, p. 77.

9. Cada lenguaje $L \in \mathbf{NP} \cap \mathbf{coNP}$ sugiere un lenguaje en **TFNP** ¿Cuál?

Christos H. Papadimitriou: "Computational Complexity", problem 10.4.16

10. Demuestre que FSAT es **FNP**-completo.

Christos H. Papadimitriou: "Computational Complexity", problem 10.4.13